ALGORITHMIC GENERATION OF AFU CALLIGRAPHY

FIELD OF THE INVENTION

The invention relates to the field of algorithmic generation of Arabic-Farsi-Urdu (AFU) script or calligraphy starting from essentially an alphabet character representation of a text.

BACKGROUND OF THE INVENTION

There are 52 languages and dialects used by over 1.9 billion people of the World that use the script of the Arabic-Farsi-Urdu (AFU) languages. The AFU languages use a cursive script with an alphabet of 28-36 characters. A character of a Middle East language like Farsi and/or Arabic will have four possible forms, namely initial, medial, final and isolated. The form of a character depends on the context of the characters, which precede and succeed it.

A method for generating the form of a character based on context sensitivity is described in "Electronic Digital System and Method for Reproducing Languages using the Arabic-Farsi Script", United States Patent No. 3,938,099 issued to Syed S. Hyder, dated February 10, 1976. It is used on all Arabic/Latin computers and Input/Output devices, and is now the International Standard ASMO 708.

In the AFU world, calligraphy is an art form. It is taught to children in elementary schools and to professional calligraphers who make it their profession. In travels in this world, one sees monuments and historical buildings adorned with classical calligraphy. There are many art museums dedicated to calligraphic writing. The rules of calligraphy as taught are well defined.

A ligature is a forced (fused) connection of two or more characters, which cannot be broken. Ligatures can be found throughout medieval hand lettering and calligraphy. Indeed, Gutenberg's font, which was an imitation of the then popular `black letter' design of northern European scribes, contained several dozen ligatures. In the rapidly evolving printing trade of the time, ligatures were not required by technology, rather they provided continuity in typographic design. It is conceivable that efficiency was a further incentive, as skilled compositors could obviously set type faster when commonly occurring letter combinations were pre-

joined. This tradition of ligature usage continued unabated until the late nineteenth century when the invention of the punch-cutting machine resulted in new composition technologies.

One of the interesting side effects of hot metal composition was that certain characters tended to break when they occurred next to one another. Ligatures were no longer a matter of aesthetics or productivity, they were essential for machine composition. It was at this time that the 'fl' and 'fi' ligatures became obligatory characters in serif type designs (and to a lesser extent in sans serif). This basic set was often extended to include 'ff', 'ffl', and 'ffi'.

As stated above, Gutenberg's invention was ideally suited to Western languages. Since they were linear, the type cast was the correct solution. The application of this technology to non-Western languages was neither natural nor correct. One would see in the case of the Chinese script, which is based on the concept of ideograms, the idea of pre-set ligatures was not feasible. In the case of Arabic-Farsi-Urdu (AFU) family of languages where the script is 2-dimensional, the idea of adapting it to fit the Gutenberg's linear print system was a wrong approach. However for reasons of domination by the West, this typecast technology was forced on the printing of the AFU script languages. As a result, a number of ligatures were invented. They were in the 4000 to 17000 ranges in various designs. It is claimed that printing good quality Arabic requires 4000 ligatures. The style of calligraphy used for Farsi-Urdu-Pashto languages, e.g. the Noori Nastalique system for Urdu, uses 17000 ligatures.

The imposition of this constraint on electronic printing of AFU languages is artificial and unjustified. The pen of a calligrapher of the AFU script follows the cultural tradition of writing. The calligrapher does not think in terms of ligatures.

In the manuscript "Authentic Arabic: a case study – 2. Technical and Aesthetic Challenges" by Thomas Milo, DecoType, Amsterdam, the Netherlands, as published in the 20th International Unicode Conference, Washington DC, January 2002, the author presents the problems associated with printing Arabic script.

Arabic calligraphy cannot be reproduced adequately and correctly with the systems that currently exist in the state of the art. Therefore, there is a need for a system that can accurately reproduce Arabic calligraphy and respect the cultural

traditions of the art.

SUMMARY OF THE INVENTION

According to a first broad aspect of the invention, there is provided a method for processing a data string of Arabic text characters into Arabic calligraphic script representation data, the method comprising: identifying words in the string; identifying a form of the characters in the words, the form comprising initial, medial, final and isolated; for the characters that are not of the isolated form, identifying a type of the characters as a function of compatibility with a type of a neighboring character; selecting, for each one of the characters in the data string, a glyph from a set of predetermined glyphs corresponding to the characters, the forms and the type; and determining a vertical offset for each glyph to match neighboring glyphs, the script representation data comprising glyph identification data and offset data for each character in the data string.

According to a second broad aspect of the invention, there is provided an apparatus for processing a data string of Arabic text characters output from an Arabic text source into Arabic calligraphic script representation data, the apparatus comprising: a word identification module receiving the data string and outputting a word; a form identification module receiving the word and outputting a form of the characters in the word, the form being one of initial, medial, final, and isolated; a type identification module receiving the form and the characters and outputting type data of the characters as a function of compatibility with a type of a neighboring character; a glyph identification module receiving the type data and the characters and selecting, for each one of the characters, a glyph from a set of predetermined glyphs corresponding to the characters, the form, and the type; and an offset determining module receiving the glyph and the characters and determining a vertical offset for the glyph to match neighboring glyphs and outputting the calligraphic script representation data.

"Arabic" is used herein to mean the family of languages using the AFU script. As mentioned above, the script of the Arabic-Farsi-Urdu (AFU) languages use a cursive script with an alphabet of 28-36 characters. A character of a Middle East language like Farsi and/or Arabic will have four possible forms, namely initial, medial, final and isolated. The form of a character depends on the position of the

character within a word. The type of a character is a subform of a form and depends on the characters which precede and succeed it. For example, the character bey, in its initial form, may have thirteen different types, or sub forms, identified as bey_initial_1,. . ., bey_initial_13. In principle this is the shape which is actually written by the calligrapher and is the graphic that is stored in the font table.

Each type is defined by a variety of attributes, which determine if a character of a certain type can be matched with the type of another character. The attributes are preferably thickness, the angle of the point of connection of a character, the direction of the pen's movement, such as horizontal, diagonal up, diagonal down, vertical, and the direction of rotation of the pen, namely clockwise, and anti-clockwise, and the waveform.

AFU script includes diacritics, similar to accent marks as used in many European languages. In the present invention, diacritics are preferably handled as independent characters and glyphs, although it is possible to provide additional glyphs representing the accented characters. Thus, an accented character can be represented by the accent character followed by the character to be accented. In AFU, more than one diacritic may be applied to a character, with the additional diacritic being added to the first diacritic. An example is the combination of the shadda diacritic and one of the tashkeel diacritics in which the tashkeel is placed over or under the shadda itself and not the letter.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features, aspects and advantages of the present invention will become better understood with regard to the following description and accompanying drawings wherein:

FIG. 1 illustrates the AFU letters of the alphabet in their four forms (initial, medial, final, and detached) and their associated names;

FIG. 2A to 2C are examples of glyph characters with varying forms and types for each glyph and their placement with respect to a point of origin;

FIG. 3A and 3B are examples of diacritics and their placement with respect to a point of origin;

FIG. 4 is a flow chart illustrating in greater detail the determination of

diacritic positioning;

FIG. 5 is a flowchart illustrating the process of attribute matching;

FIG 6 is a flow chart illustrating the process of generating script data from character data according to the preferred embodiment;

FIG 7 is an illustration of what happens on screen as characters forming a single word are typed together;

FIG. 8 is a block diagram of the apparatus according to one embodiment of the present invention; and

FIG. 9 is a block diagram illustrating the apparatus as part of a system including a user device and a printer.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

In Arabic, words are written in a cursive style, there is no discrete type of handwriting. Because of this, letters are written in four different forms depending on their location in words. Figure 1 is a table illustrating the four forms of the Arabic script for each letter: initial, medial, final, and detached. The following six letters: As={Alif, Dal, Thal, Ra', Zay, Waw} have the same medial and final form. This means that these letters cannot be joined with the letter that comes after them when they come in the middle or beginning of a word. When a letter of this group comes next to another letter from the same group, the second one is written in detached form.

Figures 2A to 2C are examples of glyph characters with various forms and types. Figure 2A is a Ha' character in its initial form and of type 1. The O indicates the point of origin with respect to which the character is placed, as well as the point at which it will be joined with a succeeding character. Since the character is in initial form, there is no point at which it is joined for a preceding character. Figure 2B is the Ra' character in its final form and in type 5. There is no joining point for a succeeding character since it is in final form. A preceding character is joined at point J. Figure 2C is the Mim character in its medial form and in type 1. This character has a joining point for a preceding and a succeeding character,

identified by P and S respectively.

Figures 3A and 3B are examples of diacritics, which are also considered to be characters by the system of the preferred embodiment. Figure 3A is a sukun. Identified on the figure are the y-displacement from the base line, the width of the glyph, the height of the glyph, and the point of origin with respect to which the glyph is placed. According to the preferred embodiment, this diacritic is considered to be a glyph by itself, without link to any other glyph with which it could be used. The sukun is placed above another glyph when used in a word. The point of origin identified in the figure is actually the highest point of the character over which it is to be placed. Once the highest point of the character is determined, the diacritic is placed above this highest point by an amount equal to the y-displacement identified in the figure. Figure 3B is a kasra, which is placed below another character. The y-displacement is the distance from the lowest point of the character underneath which it is placed. This determines the position of the diacritic.

Figure 4 is a flowchart illustrating the method by which the diacritics are placed in a word. Each diacritic is considered to be a separate glyph and does not change the position or shape of a glyph with which it is associated. In some instances, more than one diacritic is placed over one character. In this case, the highest/lowest point of a character which is used as a reference point for placement is the diacritic beneath it, not the letter.

Figure 5 is a flowchart illustrating the process of attribute matching for the method of the preferred embodiment. It corresponds to the following example.

Let {A} be the alphabet set of AFU and c_i be a character so that

$c_i \in A, \mid 1 =< i =< 35$

Let $\{A_S\}$ be a subset of {A}, being the special characters {Alif, Dal, Thal, Ra', Zay, Waw }

Let ω_j (1,...,4) be the set of forms for each character of AFU

(isolated (ISOL), initial (INIT), medial (MED), final (FIN))

Define {B} as the set of forms so that

$\omega i,j \in B, \mid 1 =< i =< 35, \mid 1 =< j =< 4,$

Let τ_k (1,...,13) be the types for each form ω for AFU

And let {T} be the set of types,

$\tau_{i,j,k} \in \{T\}$; | 1=< i =< 35, | 1=< j =< 4, | 1=< k =< 13

Define $L_{i,j,k}$ as the left attribute of the character c_i of the form ω_j and type τ_k.

Similarly $R_{i,j,k}$ be the right attribute of the character c_i of the form ω_j and type τ_k.

Next we define the value set X which corresponds to the values of the associated attribute set Φ.

Let X $(v_1,\ldots,v_n)$ be the value set of the attribute set Φ $(a_1,\ldots,a_n)$ then:

a_1 has the value v_1

...

...

...

a_n has the value v_n

If a character belongs to $\{A_S\}$=(Alif, Dal, Thal, Ra', Zay, Waw)and starts a word, then the character is considered to be isolated (ISOL).

If a character belongs to $\{A_S\}$=(Alif, Dal, Thal, Ra', Zay, Waw) and occurs in the middle of a word, then it takes on the final form (FIN), and is followed by a narrow space (like an en space in Latin languages).

For example:

$X^l_{i,j,k}$ $(v_1,\ldots,v_n)$ will be the left value set for the associated left attribute set $\Phi^l_{i,j,k}$ $(a_1,\ldots,a_n)$ for the character c_i of form ω_j and type τ_k

Like wise

$X^r_{i,j,k}$ $(v_1,\ldots,v_n)$ will be the right value set for the associated right attribute set $\Phi^r_{i,j,k}$ $(a_1,\ldots,a_n)$ for the character c_i of form ω_j and type τ_k.

Figure 6 illustrates the process of generating script data from character data. According to the preferred embodiment, a data string of Arabic text characters comprises letters, accent characters and spaces and/or other punctuation marks. To generate data faithfully representing Arabic calligraphic script, the characters are first parsed to identify words. From the position of the letters in the words, a form of the characters in the words is identified, namely as initial, medial, final and isolated or detached. For characters that are not of the isolated form, a type of the characters is identified as a function of compatibility with a type of a neighboring character. For each one of the characters, a glyph is selected from a set of predetermined glyphs corresponding to the characters,

forms and type. The glyphs are preferably designed to have a connection point (for characters to be calligraphically joined to neighboring characters) at a predetermined position within the glyph definition. Whether in a set position or not, the vertical offset for each glyph to match neighboring glyphs is determined. The script representation data thus comprises glyph identification data and, if necessary, explicit offset data, for each character in the character data string. Accents or diacritics preferably involve a specification of an offset parameter for the diacritics with respect to the letter to be accented.

A glyph is a member of a set of types and ligatures, and a font is a combination of glyphs used for printing. A synthesizer software selects the appropriate glyphs to generate the words of the AFU language as originally written by the calligrapher. The Algorithm for type definition is:

1. Perform a backward scan starting from a word separator,
2. $\omega(c_1)$ is final form,
   else repeat
3. $\omega(c_2)$ is medial or initial,
4. $\tau(\omega(c_1))$ is final ==> find $\tau(\omega(c_2))$, by pattern matching,
5. $\tau(\omega(c_2))$ ==> find $\tau(\omega(c_3))$ by pattern matching,
6. Repeat $\tau(\omega(c_i))$ ==> find $\tau(\omega(c_{i+1}))$,
7. $\tau(\omega(c_{i+1}))$ is initial terminate.

The procedure for the forward scan is outlined in figure 5.

The synthesizer considers diacritics as glyphs. In prior art systems, diacritics are stored in a reserved font space which is necessary, but combination of diacritics are considered as ligatures and occupy additional font space. In the system of the present invention, combinations of diacritics are generated by the synthesizer and do not require additional font space.

A diacritic compiler disallows unacceptable combinations of diacritics. Glyphs can be modified as required without concern to ligatures that are not used. New fonts can be easily developed as only the required glyphs are defined for a font.

The character placement must allow for the context dependent height and width positioning, so that for instance, if c_i, c_j,..., c_n are characters that link in sequence when stacked vertically, up or down, with respect to the successor or

(predecessor) so that the height of the end of a first character c_i is the beginning of the next character c_j. Likewise, if c_i, c_j,..., c_n are characters that link in sequence when they are placed horizontally, with respect to the successor or (predecessor) at different widths then the end of a first character c_i is the beginning of the next character c_j. For reasons of connectivity the characters c_i and c_j join at the appropriate height and horizontal position.

Figure 7 illustrates what happens on a screen as each character is typed on a keyboard. Each box of the table represents one character being added. What is above the dotted line represents the glyphs as they appear on the screen to form a word. What is below the dotted line are the separate characters that have been added to the word. From the sequence of boxes from 1 to 10, it can be seen that each character is modified as another character is added to it. It can also be seen that as the word becomes complete, the position of each glyph is adjusted such that true calligraphy is created in a downward vertical way. A model of real writing is created to illustrate correct Arabic text. In figure 7, the last typed character is treated as the end of the word, instead of waiting for a space or a word separator to define the end of the word, and is modified as another character is typed subsequently. The system presumes the word is complete and presents the calligraphically correct incomplete word-in-progress.

The system of the present invention is a dynamic algorithm which eliminates the increasingly lengthening look-up tables of the prior art systems. As a result of the dynamic nature of the method and system, a model of real Arabic writing is created, ligatures are unnecessary, and the footprint of the entire system is very small. While illustrated in the block diagrams as groups of discrete components communicating with each other via distinct data signal connections, it will be understood by those skilled in the art that the preferred embodiments are provided by a combination of hardware and software components, with some components being implemented by a given function or operation of a hardware or software system, and many of the data paths illustrated being implemented by data communication within a computer application or operating system. The structure illustrated is thus provided for efficiency of teaching the present preferred embodiment.

Figure 8 is a block diagram illustrating the processing apparatus 21. An

Arabic text source 20 feeds a data string into the apparatus 21, received by a word identification module 22. The word identification module 22 identifies the word in a string and feeds the word to the form identification module 24. The form identification module 24 determines if each character in the word is of initial, medial, final, or isolated form and feeds the form associated with each character to the type identification module 26. The type data and its associated character is then fed to the glyph identification module 28, where a glyph is selected from a set of predetermined glyphs that corresponds to the type and form for each character. The glyph and all its relevant information is input into the offset determining module 30, where the glyph and offset are combined to produce and output the calligraphic script representation data. This data can be sent to a database 32 for storage, to a display module for display on a screen (not shown), or directly to a printing device for printing onto paper.

In a preferred embodiment, the type identification module 26 identifies a best match of attributes between glyphs available in a set of glyphs for a form of a character, the best match corresponding to a visualization of a calligrapher. Also in a preferred embodiment, the word identification module 22 identifies diacritics as separate characters in the string and associates the diacritics to separate glyphs in the set of predetermined glyphs. The offset determining module 30 determines an offset position of each diacritic to be associated with a glyph representing a letter. The word identification module 22 verifies unacceptable combinations of diacritics disallows them.

Figure 9 illustrates the different emplacements for the processing apparatus 21. In a user device 36, such as a computer, comprising an application 38 and a printer device driver 40, the processing apparatus may be comprised as software within the printer device driver 40. Alternatively, it may be in the printer 34, or any type of electronic printing apparatus. A typical printer including an input/output interface 42, an image/text translator 44, a printing controller 46, and a mechanical printing device 48 can include the processing apparatus 21 at various locations. It can be before the input/output interface 42 and input the calligraphic script representation data directly into the input/output interface 42. It can be before the image/text translator 44 and input the calligraphic script representation data into that module, or it can be an integral part of the image/text translator

module 44 and input the data directly to the printing controller 46. In certain printers, the image/text translator module 44 is an equivalent to a page definition language compiler. The present invention can be an extension to Postscript™ as it can output to the printing controller 46 the same type of information as if a page definition language were used. All the information required for the printer to place the glyphs on the page are present, namely glyphs (including form and type) and offsets.

Alternatively, the processing apparatus 21 can be a plug-in to an internet browser. It can be a web browser comprising a translator that takes standard HTML text and converts it onscreen to calligraphic script representation data.

It should be noted that the present invention can be carried out as a method, can be embodied in a system, a computer readable medium or an electrical or electro-magnetic signal.

It will be understood that numerous modifications thereto will appear to those skilled in the art. Accordingly, the above description and accompanying drawings should be taken as illustrative of the invention and not in a limiting sense. It will further be understood that it is intended to cover any variations, uses, or adaptations of the invention following, in general, the principles of the invention and including such departures from the present disclosure as come within known or customary practice within the art to which the invention pertains and as may be applied to the essential features herein before set forth, and as follows in the scope of the appended claims.